# PCI-SIG ENGINEERING CHANGE REQUEST

| | |
|---|---|
| **TITLE:** | Downstream Port Containment related Enhancements |
| **DATE:** | Draft version – September 10, 2018 |
| **AFFECTED DOCUMENT:** | PCI Firmware Specification, Rev. 3.2 |
| **SPONSOR:** | Mahesh Natu, Rob Gough (Intel Corporation) |

## Part I

### 1. Summary of the Functional Changes

Changes are requested to be made to Section 4.5.1, _OSC Interface for PCI Host Bridge Devices, Section 4.5.2.4 Dependencies Between _OSC Control Bits. Section 4.6.12 and 4.6.13 are newly added.

The changes will enable the Operating System to advertise its Downstream Port containment related capabilities to the firmware. It also enables the Operating System and the Firmware to negotiate ownership of Downstream Port Containment extended capability register block and collaboratively manage Downstream Port Containment events.

### 2. Benefits as a Result of the Changes

PCIe Base Specification suggests that Downstream Port Containment may be controlled either by the Firmware or the Operating System. It also suggests that the Firmware retain ownership of Downstream Port Containment if it also owns AER. When the Firmware owns Downstream Port Containment, it is expected to use the new “Error Disconnect Recover” notification to alert OSPM of a Downstream Port Containment event. This PCI Firmware specification change enables the firmware to discover whether the Operating System supports Downstream Port Containment handling and allow such an Operating System and the Firmware to establish ownership of Downstream Port Containment register block.

Here is the implementation note from PCIe 4.0 Specification for reference -

DPC may be controlled in some configurations by platform firmware and in other configurations by the operating system. DPC functionality is strongly linked with the functionality in Advanced Error Reporting. To avoid conflicts over whether platform firmware

or the operating system have control of DPC, it is recommended that platform firmware and operating systems always link the control of DPC to the control of Advanced Error Reporting.

## 3. Assessment of the Impact

Table 4-4, Table 4-5 and Table 4-6 in Section 4.5.1 must be extended to add Downstream Port Containment related Support and Control bits. Section 4.5.2.4 must be extended to describe the interdependencies between advanced error reporting control bit and downstream port containment control bit. New sections, specifically section 4.6.12 and 4.6.13, are added to describe related DSM functions.

## 4. Analysis of the Hardware Implications

No impact.

## 5. Analysis of the Software Implications

These software changes are optional.

The Operating System needs to add support for Error Disconnect Recover notification on a PCIe Root Port, PCIe downstream switch port or a PCIe Host Bridge device in ACPI namespace. In addition, the Operating system advertises support for this by setting bit 7 of _OSC Support Field. This enables a mode of operation where the firmware owns both Advanced Error Reporting and Downstream Port Containment.

The Operating System needs to add support for native handling of Downstream Port Containment events and negotiate control of Downstream Port Containment with the Firmware via bit 7 of _OSC Control Field. Firmware can choose to retain control of Downstream Port Containment along with Advanced Error Reporting. Alternately, the Firmware may choose to grant ownership of Downstream Port Containment and Advanced Error Reporting to the Operating System.

When firmware controls configuration of Downstream Port Containment Extended Capability, the operating system needs to support invocation of two additional DSM methods – one that enables/disables Downstream Port Containment and one that is used to locate the port that was affected by a containment event.

## 6. Analysis of the C&I Test Implications

No impact.

## Part II

**Detailed Description of the change**

*Change Section 4.5.1 Table 4-4 as follows:*

**Table 4-4 Interpretation of _OSC Support Field**

<table>
<tr><th>Support Field bit offset</th><th>Interpretation</th></tr>
<tr><td>..</td><td>..</td></tr>
<tr><td>7</td><td>Error Disconnect Recover Supported<br><br>The OS sets this bit to 1 if it supports Error Disconnect Recover notification on PCI Express Host Bridges, Root Ports and Switch Downstream Ports. Otherwise, the OS sets this bit to 0. See ACPI specification for definition of Error Disconnect Recover Notification.<br><br>In the context of PCIe, support for Error Disconnect Recover implies that the OS will invalidate the software state associated with child devices of the port without attempting to access the child device hardware. If the OS supports Downstream Port Containment (DPC), as indicated by the OS setting bit 7 of _OSC control field, the OS shall attempt to recover the child devices if the port implements the Downstream Port Containment Extended Capability. If the OS continues operation, the OS must inform the Firmware of the status of the recovery operation via the _OST method. The upper word of Arg1 argument of _OST carries the Bus, Device, and Function number of the port that experienced the containment event. The segment number of this port is equal to the segment number of the Error Disconnect Recover notification target. The layout of Arg1 is as follows<br><br>
<table>
<tr><th>Bit Position</th><th>Definition</th></tr>
<tr><td>15:0</td><td>Status of the operation. See ACPI specification.</td></tr>
<tr><td>18:16</td><td>Function number of the port that experienced the containment event</td></tr>
</table>
</td></tr>
</table>

| | | |
|---|---|---|
| | 23:19 | Device number of the port that experienced the containment event |
| | 31:24 | Bus number of the port that experienced the containment event |

*Change Section 4.5.1 Table 4-5 as follows:*

**Table 4-5 Interpretation of _OSC Control Field, Passed in via Arg3**

| **Support Field bit offset** | **Interpretation** |
|---|---|
| .. | **..** |
| 7 | **PCI Express Downstream Port Containment Configuration control**<br><br>The OS sets this bit to 1 to request control over configuration of PCI Express Downstream Port Containment Configuration. If the OS successfully receives control of this feature, it must properly configure Downstream Port Containment control registers and handle Downstream Port Containment events as described in the PCI Express Base Specification; further, the operating system retains control of Downstream Port Containment across power transitions for S1, S2, S3 system power states. If this bit is set by the OS, this indicates that it supports both native OS control and firmware ownership models (i.e. Error Disconnect Recover notification) of Downstream Port Containment. |

*Change Section 4.5.1 Table 4-6 as follows:*

**Table 4-6 Interpretation of _OSC Control Field, Returned Value**

| Support Field bit offset | Interpretation |
|---|---|
| .. | .. |
| 7 | **PCI Express Downstream Port Containment configuration control**<br><br>Firmware sets this bit to 1 to grant the OS control over PCI Express Downstream Port Containment configuration. If firmware allows the OS control of this feature, then, in the context of the _OSC method the OS must ensure that Downstream Port Containment ERR_COR signaling is disabled as described in the PCI Express Base Specification. Additionally, after control is transferred to the OS, firmware must not modify any registers in the Downstream Port Containment Extended Capability Structure. Any firmware configuration of the Downstream Port Containment Capability may be overwritten by the OS, depending on OS policy.<br><br>If control of this feature was requested and denied, or was not requested, the firmware returns this bit set to 0.<br><br>If control of this feature was requested and denied, firmware is responsible for initializing Downstream Port Containment Extended Capability Structures per firmware policy. Further, the OS is permitted to read or write DPC Control and Status registers of a *port* while processing an Error Disconnect Recover notification from firmware on that port. Error Disconnect Recover notification processing begins with the Error Disconnect Recover notify from Firmware, and ends when the OS releases DPC by clearing the DPC Trigger Status bit. Firmware can read DPC Trigger Status bit to determine the ownership of DPC Control and Status registers. Firmware is not permitted to write to DPC Control and Status registers if DPC Trigger Status is set i.e. the link is in DPC state. Outside of the Error Disconnect Recover notification processing window, the OS is not permitted to modify DPC Control or Status registers; only firmware is allowed to. |

*Change Section 4.5.2.4 as follows:*

### 4.5.2.4. Dependencies Between _OSC Control Bits

Because handling of hot-plug events, power management events, and advanced error reporting all require the modification of PCI Express Capability registers, the operating system is required to claim control over the PCI Express Capability (bit 4 of the Control field) in conjunction with claiming control over PCI Express Native Hot Plug, PCI Express Native Power Management Events, or PCI Express Advanced Error Reporting (bits 0, 2, and 3 of the Control field). If the operating system attempts to claim control of any of these features without also claiming control over the PCI Express Capability, the firmware is required to refuse control of the feature being illegally claimed and mask the corresponding bit.

Because handling of Downstream Port Containment has a dependency on Advanced Error Reporting, the operating system is required to request control over Advanced Error Reporting (bit 3 of the Control field) while requesting control over Downstream Port Containment Configuration (bit 7 of the Control field). If the operating system attempts to claim control of Downstream Port Containment Configuration without also claiming control over Advanced Error Reporting, firmware is required to refuse control of the feature being illegally claimed and mask the corresponding bit. Firmware is required to maintain ownership of Advanced Error Reporting if it retains ownership of Downstream Port Containment Configuration.

If the OS sets bit 7 of the Control field, it must set bit 7 of the Support field, indicating support for the *Error Disconnect Recover* event.

Operating systems must comprehend that platforms may not grant control of the Native PCI Express Advanced Error Reporting feature and Native PCI Express Downstream Port Containment Configuration feature.

*Update Table 4-7 in Section 4.6 as follows:*

**4.6 _DSM definitions for PCI**

_DSM (Device Specific Method) is defined in the ACPI 3.0 (or later) specification. This object is a control method that enables devices to provide device specific control functions that are consumed by the device driver. Table 4-7 below list the UUID, revision and function definitions.

Table 4-7: _DSM Definitions for PCI

| UUID | Revision | Function | Description |
|---|---|---|---|
| E5C937D0-3553-4d7a-9117-EA4D19C3434D | .. | .. | |
| | 5 | 0Ch | Downstream Port Containment Enable Request from OS |
| | 5 | 0Dh | Locate the port that experienced the containment event |
| | | | |

*Add Section 4.6.12 as follows:*

## 4.6.12 _DSM for Downstream Port Containment Enable Request from OS

This section describes how the operating system can request the firmware to enable DPC for a given DPC capable root port or a switch port when the firmware controls DPC. The request is communicated through the _DSM ACPI method. This _DSM function is optional. This method, if present, must be evaluated by the DPC aware OS at least once after the _OSC handshake if firmware retains control of DPC. As a result of plug and play operations, this method may be evaluated multiple times during the OS operation with differing inputs. If the OS has control of DPC, this method is not evaluated by the OS. The OS is not permitted to evaluate _DSM while the port is in DPC triggered state.

If this method is not implemented and the firmware owns DPC, the firmware is solely responsible for determining whether DPC should be enabled or disabled.

**Location:**
This object can be placed under any object representing a DPC capable PCI Express Root Port or Switch Downstream Port. Children of a device inherit from their parent unless they contain an overriding _DSM with this function ID.

**Arguments:**
Arg0: UUID: E5C937D0-3553-4D7A-9117-EA4D19C3434D
Arg1: Revision ID: 5
Arg2: Function Index: 0Ch
Arg3: Downstream Port Containment Enable request from OS

Type: Integer

Description:
0: The OS is requesting that firmware should keep DPC disabled. In response to this request, the firmware shall disable DPC.
1: The OS is requesting that firmware should enable DPC. In response, the firmware may enable DPC or choose to keep it disabled.

**Return:** Downstream Port Containment Status from firmware

Type: Integer

Description:
0: DPC is disabled.
1: DPC is enabled.

*Add Section 4.6.13 as follows:*

## 4.6.13. _DSM for locating the port that experienced the containment event

Firmware may wish to issue *Error Disconnect Recover* notification on a port that is parent of the port that experienced the containment event. This can be facilitated by implementing this DSM under the parent port scope. This method, if present, must be evaluated by the DPC aware OS after *Error Disconnect Recover* notification. This method returns the bus, device and function number of the child port where the

containment event occurred. If the parent is a root port, the return value may represent the bus, device and function number of a switch downstream port that entered containment mode. The segment number of the child port is assumed to be identical to the parent port.

If this DSM is not implemented under the port that is the target of *Error Disconnect Recover*, the OS assumes that the target of the notify event is the port that experienced the containment event.

**Location:**
This object can be placed under any object representing a DPC capable PCI Express Root Port or Switch Downstream Port. If a port implements this DSM, its child devices cannot instantiate this DSM function.

**Arguments:**
Arg0: UUID: E5C937D0-3553-4D7A-9117-EA4D19C3434D
Arg1: Revision ID: 5
Arg2: Function Index: 0Dh
Arg3: 0

**Return:** Location of the port that experienced containment event

Type: Integer

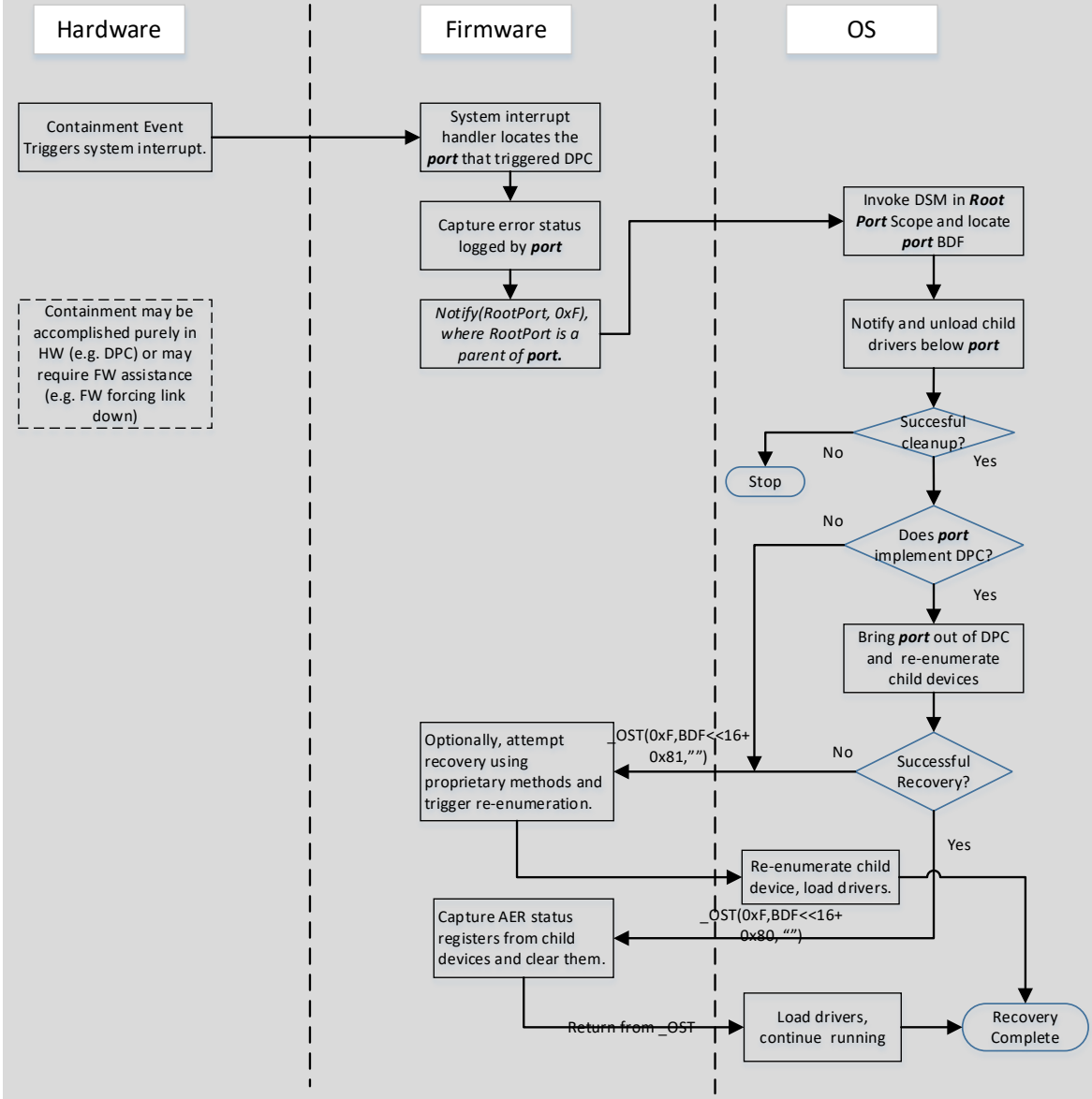
Description:

Port Bus, Device, and Function number encoded as a 16 bit quality.

(Bits 2:0 = Function, Bits 7:3 = Device, Bits 15:8 = Bus)

## IMPLEMENTATION NOTE

### DPC event handling when firmware maintains control of AER and DPC



**Sample ASL code**

```
Scope(_SB){
Device(PCI0) {   //
      Name (_HID, EISAID("PNP0A03"))
      Name (_UID, 0)
       //..
     Method(_OSC,4, Serialized) {
      // Extended to support DPC specific extensions
     }

      Device(RP0) { // Root port 0, supports DPC
          Name (_ADR , 0)  // Device 0, function 0

          OperationRegion (DPC0, PCI_Config, 0x160, 0x20)
          Field (DPC0, AnyAcc, NoLock, Preserve)
          {
             Hdr, 32 // Start of DPC register block
             // define rest of the DPC registers so that FW can access these from ASL
           }

        Method (_DSM, 4, Serialized)
        {
           // check for GUID and revision match
         Switch(Arg2) {
          Case(0xC) {
                // Enable/disable DPC request, replace x with the spec assigned value
             If (LEqual(Arg3, 0)) {
                //disable DPC by modifying registers in DPC0 op region
             }
             If (LEqual(Arg3, 1)) {
               //Enable DPC by modifying registers in DPC0 op region
             }
           } //end Arg2=0xC
         Case (0xD) {
```

```
            // return BDF of port that experienced containment
          } // end Arg2=0xD
        } // end DSM

        Method(_OST, 3, Serialized) {
          // OSPM calls this method after processing ErrorDisconnectRecover notification from firmware
          Switch(And(Arg0,0xFF)) // Mask to retain low byte
           {
            Case(0x0F) { // Error Disconnect Recover request
               Switch(AND(Arg1, 0xFF))  {
                 Case (0x80)  {// Success
                     // Extract BDF of the port with containment event from upper word of Arg1.
                     // Read AER status register from the endpoint below this port.
                      } // end Case(0)

                 Default {
                   //IO recovery failed or unrecognized status code
                   // Optionally, attempt FW specific recovery.  OK to do nothing
                 } // End Default

             } // End Case(0xF)
           } // End Switch
        } // End _OST
      } // End RP0
    //..
  } // End PCI0
} // End SB
```